



# Using Rulesets to Build and Manage Data

Todd A. King, Steven P. Joy, Joseph N. Mafi, Erin K. Means, Raymond J. Walker  
Institute of Geophysics and Planetary Physics, UCLA, 3846 Slichter Hall, Los Angeles, CA 90095-1567 United States

## Purpose and Goal

- To create an efficient, platform independent language for the collection and formatting of metadata.
- To improve and enhance coordination between data engineers and data providers.
- To formalize and capture the “business” rules used in the generation of archive data sets.

## Why Rulesets

- Data engineers used different programming languages and approaches to design and process archival data.
- Sharing and re-use of software was difficult or non-existent.
- Portability between platforms was limited.
- Logic (rules) were obscured by code.

## What is a Ruleset

- A ruleset is a collection of one or more rules.
- A rule is a statement of action (i.e, assign a value to a variable, include another ruleset, run an external application, write output, display a message).
- Flow through a ruleset may be conditional. (Support for IF/ELSEIF/ELSE)

## How Rulesets Work

- A ruleset is applied to a specific file.
- Metadata about the file (name, extension, size, time stamp, location) can be used to control ruleset execution.
- Rulesets can call (include) other rulesets and external applications (plug-ins) based on metadata.
- Output is generated by replacing variable tokens in a template with the current value of a variable.

## Power of Ruleset

- Allows a modular design. (Large processing tasks can be divided into smaller, more tailored rulesets)
- Sharing and re-use is common (plug-ins leverage specialized applications more effectively)
- Data engineers can develop rulesets that can readily be used by data providers.
- Rulesets capture (business) logic.
- Stream-lined “language” tailored for data preparation and metadata collection.

## Ruleset Language

### Comments

A comment is any line of text that begins with either “#” or a “/” or text enclosed between “/\*” and “\*/”.

### Variables

A variable is a named value. Values may be strings, literals, or arrays.

### Directives

Directives are commands to the ruleset processor which control which rules are executed and provide an interface to external files or applications (plug-ins) for acquiring rulesets.

### OPTION

The OPTION directive sets the value of an option for the ruleset processor.

### GLOBAL

The GLOBAL directive defines a variable that should persist between executions of individual rulesets.

### INCLUDE

The INCLUDE directive instructs the ruleset processor to open a file and load the contents as a set of rules.

### MESSAGE

The MESSAGE directive provides a means to display a message for the user. A message may span multiple lines.

### IF

The IF directive marks the beginning of a block of rules which will be executed if the value associated with a variable matches the specified pattern.

### /IF

The /IF directive marks the end of the block of rules that was marked with the most recent IF directive.

### ELSE

The ELSE directive marks the beginning of a block of rules which will be executed if the conditions of the preceding IF directive are not met.

### ELSEIF

The ELSEIF directive marks the beginning of a block of rules which will be executed if the value associated with a variable matches the specified pattern.

### ABORT

The ABORT directive ends the processing of the rules and reports that all processing should end.

### IGNORE

The IGNORE directive ends the processing of the rules and reports that no output should be generated.

### TEMPLATE

The TEMPLATE directive defines the file which will be used generating output.

### COPY

The COPY directive instructs the ruleset processor to copy a file from one location to another.

### OUTPUT

The OUTPUT directive defines the name of the file the output will be written.

### RUN

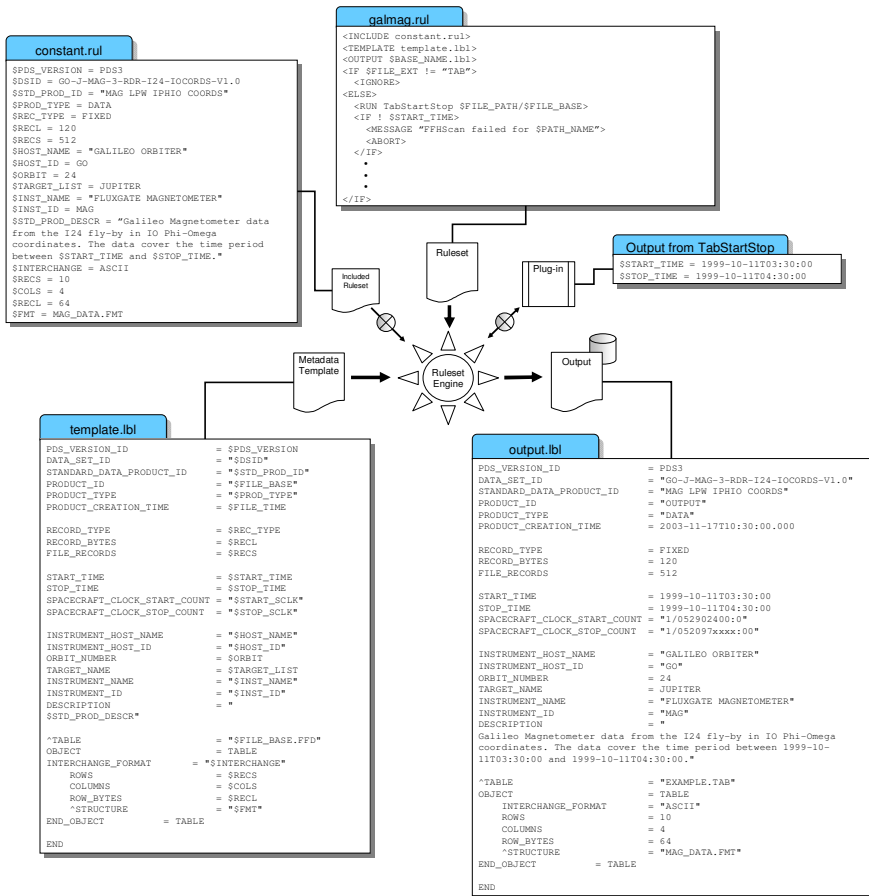
The RUN directive will execute a command, passing any number of arguments, and process the output from the command as a set of rules.

## Example

Generate PDS labels for all Galileo Magnetometer data sets for orbit 24. These data are stored as ASCII tables (files have the extension of “TAB”). All common parameters are stored in “constants.rul” and included by the master ruleset “galmag.rul”. The plug-in “TabStartStop” is used to determine start and stop times from the data file. The template label is stored in “template.lbl”. Output is placed in a file with the same base name as the data file with the extension of “LBL”.

An entire directory tree of files is processed with a command like:

```
labeler galmag.rul /galileo/i24
```



## Actual Implementation of a Ruleset Processor

```
labeler
import pds.ruleset.*;
import java.io.*;
import java.util.*;

/* Loads a rule set and processes one or more files. */
class Labeler {
    /** Preserves global variables between executions of the ruleset. */
    public static ArrayList mGlobalList = new ArrayList();

    /* Create an instance. */
    public Labeler() {
    }

    /* Entry point for the application. */
    public static void main(String[] args) {
        // Check arguments
        if(args.length < 2) {
            System.out.println("Usage: labeler ruleset pathname [pathname ...]");
            return;
        }

        // Process arguments
        for(int i = 1; i < args.length; i++) {
            if(!processItem(args[i], args[i])) break;
        }

        static boolean processItem(String ruleset, String pathName) {
            File item = new File(pathName);

            if(item.isDirectory()) {
                File[] list = item.listFiles();

                for(int i = 0; i < list.length; i++) {
                    if(list[i].isDirectory()) {
                        processItem(ruleset, list[i].getPath());
                    } else {
                        if(!runRuleset(ruleset, list[i].getPath())) return false;
                    }
                }
                return true;
            }
            return runRuleset(ruleset, pathName);
        }

        static boolean runRuleset(String ruleFile, String pathName) {
            PPIRuleset ruleset = new PPIRuleset();
            boolean good = false;

            // Set global variables
            for(int i = 0; i < mGlobalList.size(); i++) {
                ruleset.mGlobalList.add(PPIVariable mGlobalList.get(i));
            }

            if(!ruleset.parse(ruleFile)) {
                ruleset.showMessage(false, "An error occurred while parsing the ruleset.");
                return false;
            }

            // Run the ruleset
            if(!ruleset.run(pathName)) {
                ruleset.showMessage(false, "One or more errors occurred while processing file: " + pathName);
                ruleset.showMessage(false, "No output file was created.");
                return false;
            }

            if(ruleset.mWriteOutput) {
                if(ruleset.update()) { // Update template
                    ruleset.output(); // Write the template out to PPI standards
                }
            }

            // Save global variables
            mGlobalList.clear();
            for(int i = 0; i < ruleset.mGlobalList.size(); i++) {
                mGlobalList.add(PPIVariable ruleset.mGlobalList.get(i));
            }

            return true;
        }
    }
}
```

## How Plug-ins Work

- Plug-ins can be written in any programming or script language.
- A plug-in must accept command line arguments.
- Input into the plug-in is specified in the RUN directive.
- Output from the plug-in is written to standard out.
- Output must be in the ruleset language.
- The output is processed in the same manner as an INCLUDE (run in separate engine with current variable transferred).

## Current Plug-ins

### CassiniFFHScan

Extract information from a Cassini Flatfile.

### Compare

Perform a relational compare two strings or numbers.

### FFHScan

Extract information from a Flatfile.

### FormatDescription

Word wrap and indent text.

### IMath

Perform simple integer math.

### LabelValue

Extract a value from a label.

### Lookup

Find a value in an interval lookup spreadsheet.

### SpreadSheet

Parse files containing a spreadsheet (delimited text) and determine metrics.

### PChronos

Interface to the NAIF/SPICE "chronos" utility.

### Strings

Determine length, change case, index, and subset strings.

### TabStartStop

Return a portion (column) of the first and last rows in an ASCII table.

### TargetPhrase

Create a properly punctuated phrase describing a list of values.

### Time

Parse and construct time strings in many formats.

## Implementation Details

- Written in Java
- Organized as a set of classes:
  - PDSLabel**: Parsing PDS Labels.
  - PPIOption**: Option handling support.
  - PPIRuleset**: PPI Ruleset Language processing.
  - PPITable**: Reading and writing to tables.
  - PPITime**: Manipulate time strings.
- Custom packages containing rulesets, plug-in and applications are created using a java based self installer.

## Actual and Potential Applications

- Generating PDS labels for new data.
- Converting old PDS labels to new standards.
- Building data distributions.

## Going Further

- Extend ruleset processing to text files (relax requirement that template be in PDS label format)
- Add geometry engine plug-in (translate between ephemeris values)
- Adapt ruleset processing for the generation of distribution values (extend COPY directive to work on directory trees).

## Summary

Using rulesets has increased our productivity. It has allowed data teams to work more effectively and has helped improve the coordination of parallel and distributed activities. We have successfully used rulesets in the preparation of Galileo fields and particles data for archiving in PDS. Also the Cassini project is using rulesets to prepare data at the science teams for delivery to the PDS. Data engineers at the PPI node work closely with the Cassini teams to design and develop the rulesets so that data deliveries meet system requirements. We plan to use rulesets to migrate data already in the PDS archives to the next generation PPI data system. We anticipate that once the rulesets are developed we can migrate our entire data archive on the order of days.

## For More Information and Downloads

<http://www.igpp.ucla.edu/pds>